A General Purpose Sparse Matrix Parallel Solver

Hong Q. Dings.ndR,ol)crt D. Ferraro
Jet Propulsion Laboratory, MS 169-315
California Institute of Technology
Pasadena, CA9I109
Email: hding@redwood.jpl.nasa.gov

Abstract

A general purpose sparse matrix parallel solver is developed primarily for solving a range of linear symmetric systems arising from discretization of partial differential equations on unstructured meshs. The key point is that sparse matrices arising from finite clement and finite difference methods can be easily cast into a form with certain storage structure and communication pattern which can be standardized in a set of subroutines. The package includes a set of subroutines for aiding users to do parallel 1/0 to read in the mesh file, partition file, to allocate storage for the sparse matrix and assemble the sparse matrix elements. The package provides a set of parallel linear solvers, such as preconditioned bi-conjugate gradient, the Cholesky fact orizat ion, quasi minimal residual solver, or a t wo-step hybrid method using an incomplete Cholesky factorization to eliminate the interior grid points and using a conjugate gradient or the Cholesky factorization to solve the resulting much smaller system consisting of grit] points along the processor boundaries, Scaling tests on Intel Delta up to 128 processors show that the bi-conjugate gradient method scales linearly with the number of processors, whereas the two step hybrid method scales with the square root of the number of processors.

Introduction

One of the common problems in scientific and engineering computations is the construction and solution of sparse coefficient matrices of linear systems arising from solving partial differential equations based on unst ruct ured grids [1]. Many such examples occur in analysis by finite clement, finite difference and finite volume methods. As the scale and complexity of calculations grow, massively parallel computers arc increasing used in these calculations. Here, wc describe a software package for handling these (symmetric) sparse matrices on a massively parallel distributed memory computer. The package includes routines to handle the compact storage scheme to store the nonzero elements of the sparse coefficient matrix, routines to construct the sparse matrix, and a choice of several possible method to solve the resulting linear equations. Fig.] illustrates the basic data flow chart of the solver package.

Domain decomposition

Subdomains of the unstructured grids arc divided among processors, The subdomain boundaries always cut through the links connecting grids, Grid points sitting on the domain boundaries arc shared among the processors whose subdomain bounders it. in the

finite element language (from which the present solver package is evolved), each element belongs to only Onc processor, but a given nodal point may physically resided on the several processors which share it,. In this case, the variables associated with the same point arc present in all those processors. Fig.2 indicates how the matrix element are splitted into pieces among the relevant processors. This matrix decomposition differs from many other decompositions where each matrix element belongs to a unique processor. The most clear advantage of such a decomposition is that the matrix-vector product is easily carried out, Wc first do a local VI = M . V., since all entries arc locally stored. This proceeds as just like on a sequential computer. Afterwards, the only entry in VI which arc not complete arc those for the boundary points, which can be completed by summing up all contributions from the relevant processors which share this point. This step is called globalize(). Fig.3 sketches this matrix-vector product in more details. This decomposition has been described previously[2-4] within the context of the finite element method, Wc emphasize that this decomposition arc also suitable for the coefficient matrices arising from a finite difference method. In fact, the application example given at the end of this paper is a finite difference method solution.

Sparse matrix storage and construction

Wc used two compact storage schemes for sparse matrices. The first onc store only nonzero matrix elements. This minimum storage scheme is convenient for the matrix-vector product usd in conjugate gradient type of iterative methods. Other other is so-called skyline profile or variable banded scheme, .i.e., on each row, wc store the first nonzero element and every element between the first nonzero element and the diagonal element. This scheme is very convenient for Cholesky fact orizat ion, alt bough the st orage is not really optimum: some zero elements are also stored in this scheme. Depending upon the solution method invoked to solve the linear equations, the package chose onc scheme or a mix of the above schemes, as discussed below.

Solution Methods

The package includes several solution methods that a user may choose to best fit the problem at hand. 'These methods include a preconditioned bi-conjugate gradient (PBCG) method which can deal with both real and complex symmetric matrices; a quasi-minimum residue (QMR) method to hand the symmetric complex matrices; a two-step hybrid method which first do a sequential Choleksy factorization to eliminate the interior points and then solve the resulting (mush smaller) reduced linear systems using either a parallel bi-conjugate gradient iteration method or a parallel Cholesky factorization.

Conjugate gradient type iterative methods (include the QMR) arc particularly suited to Solve linear equations on parallel computers because they only involves matrix-vector product and global sums. Matrix-vector product is easily carried out in our data dccon~-

Performance

We have completed the solver package. The solver package is applied both to a finite element solution of electromagnetic wave scattering from conducting sphere, and to a finite difference solution to a static heat distribution problem governed by Poisson equation.

In Fig.4, we plotted the scaling behavior of the solver for the finite difference solution to a Poisson equation. We let the problem size scale with number of processors while fixing 1600 grid points per processor, The data shown is the ratio of the time for solving the entire problem on M processors vs. the time for solving a 1600-grid problem on one processor. The preconditioned bi-conjugate gradient (PBCG) methods scales as N^2, where one N comes from the matrix-vector product (sparsity reduces N^2 to N) and another N comes from the order N iterations. Since $N \sim M$, we expect the scaling be linear in M, and our data clearly supports this scaling behavior. The two-step hybrid first dots a sequential incomplete factorization on the interior points and then dots a parallel bi-conjugate gradient method for those boundary points. The first part remains a constant for the fixed size per processor, and the second part deals with those boundary points which increase as square root of the total points. Once the first part saturates, the total time for the hybird method should increase as $\sqrt{M}$, as our data indicated.

Conclusion

We have implemented a software package for constructing and solving symmetric sparse coefficient matrix of linear systems arising from solving partial differential equations based on unstructured grids. The interface of the solver is through a series of subroutine calls, There are several solution methods provided for user to chose. The scaling tests indicate good scaling behavior of the two-step hybrid conjugate gradient - Cholesky method.

This work is funded under a NASA HPCC ESS contract.

References:

1 1.S. Duff, A.M. Erisman, and J.K. Reid. *Direct Methods for Sparse Matrices,* Oxford University Press, London, 1986.

2 G.A. Lyzenga, A. Raefsky and B. Nour-Omid, "Implement Finite Element Software on Hypercube Machines", in *The Third Conference on Hypercube Concurrent Computers and Applications,* Ed.G.C. *Fox,* p.1755, VOI.2, 1988, ACM Press, New York.

3 *Solving Problems* 071 *Concurrent Processors*, G.C. Fox, M.A. Johnson, G.A.Lyzenga, S.W. Otto, J.K. Salmon and D.W. Walker, Chap.8, Vol.], Prentice Hall, Englewood Cliffs, New Jersey, 1988.

4 J .W. Parker, R.D. Ferraro and P.C. Liewer, "Comparing 3D Finite Element Formulations Modeling Scattering from a Conducting Sphere", IEEE Trans. on Magnetics, p.1646, VO1.29, No.2, March 1993.

Fig. 1

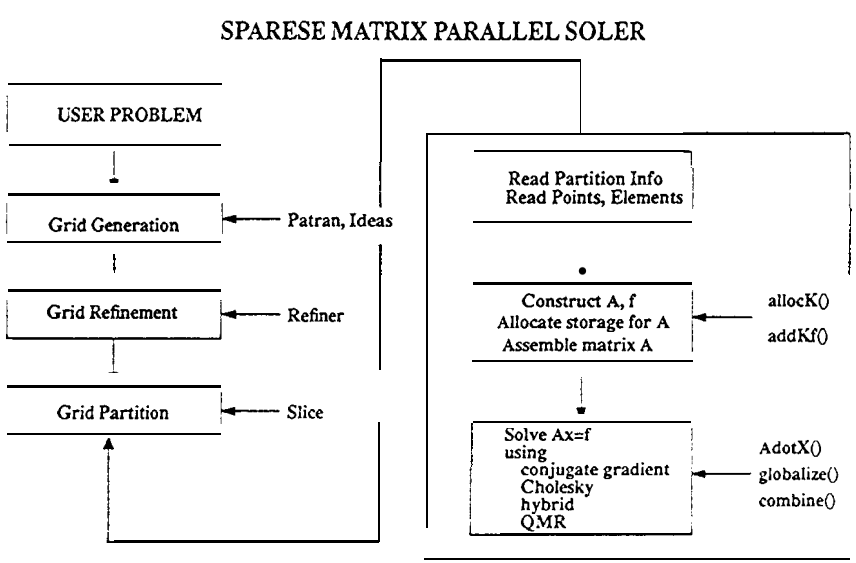
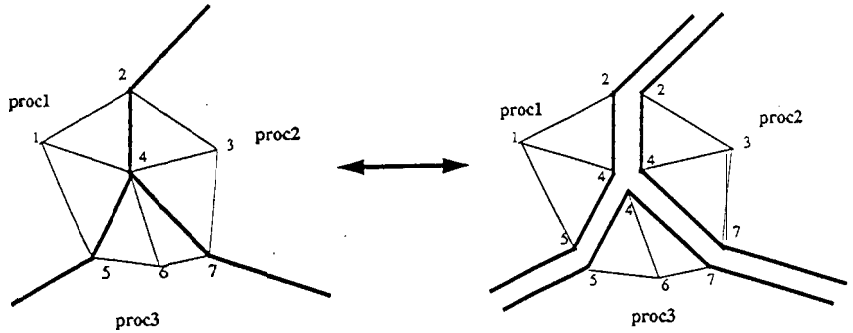


Fig. 2

Breakup of the Sparse Matrix A for Finite Elements



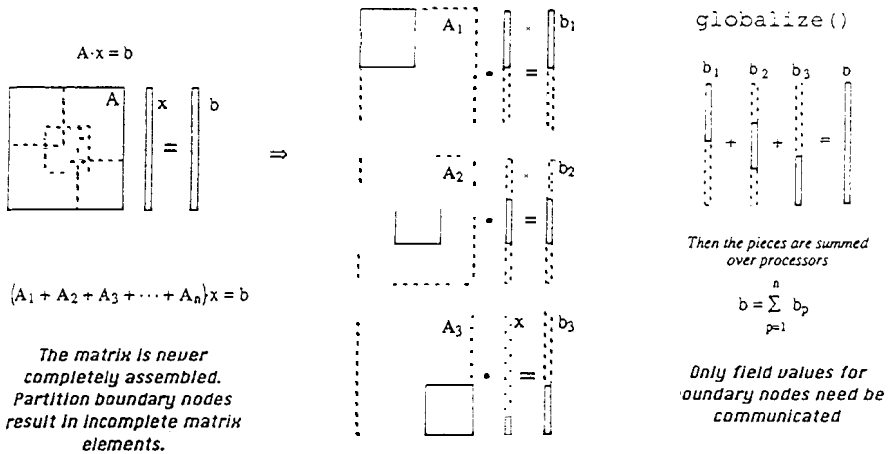
$$A_{44} = \overbrace{A_{44}^{(124)} + A_{44}^{(145)}}^{\text{Proc1}} + \overbrace{A_{44}^{(234)} + A_{44}^{(347)}}^{\text{Proc2}} + \overbrace{A_{44}^{(476)} + A_{44}^{(465)}}^{\text{Proc3}}$$

$$A_{24} = A_{24}^{(124)} + A_{24}^{(234)} + 0$$

Fig. 3

Global View of the Linear Solver

parallel matrix - vector multiply, vector dot products



$(A_1 + A_2 + A_3 + \cdots + A_n)x = b$

The matrix is never completely assembled. Partition boundary nodes result in incomplete matrix elements.

Then the pieces are summed over processors

$$b = \sum_{p=1}^{n} b_p$$

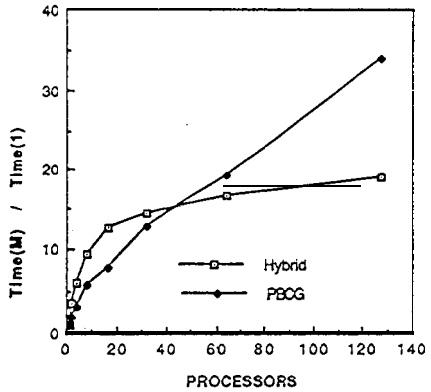
Only field values for boundary nodes need be communicated

matrix multiply in parallel

$A_1 \cdot x = b_1,\ A_2 \cdot x = b_2,\ A_3 \cdot x = b_3,\ \cdots,\ A_n \cdot x = b_n.$

Fig. 4

of Sparse Matrix Parallel Solver on Intel Delta



The problem size scales with the number of processors such that each processor contains fixed 1600 nodal grids. T(M) is the time for solving a 1600M grid problem on M processors. T(1) is the time for solving a 1600 grid problem on one processor. The pre-conditioned conjugate gradient method scales linearly as expected. The hybrid method scales as square root of M, which is superior to PBCG for large problems.